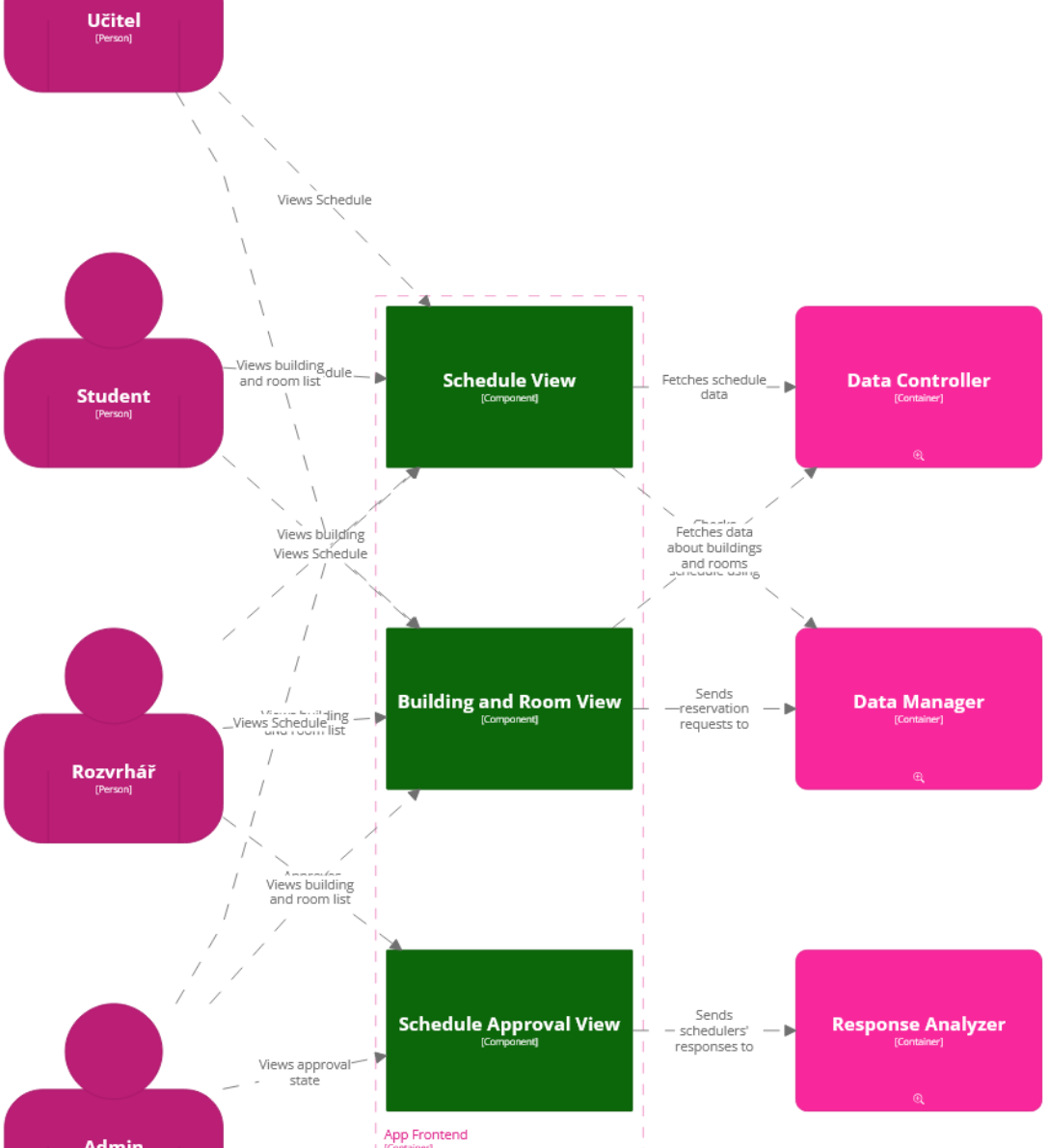
ENR1 – SCH1 Quality Scenarios

Milestone 2, NSWI130, 2024

ENR1 – Matyáš Černíček, Jan Čelovský, Šimon Jůza, Jan Růžička, Vojtěch Snop, Rastislav Vojtuš

App_Frontend



Quality requirement scenario 1 – App Frontend

Run-time dimension - Security

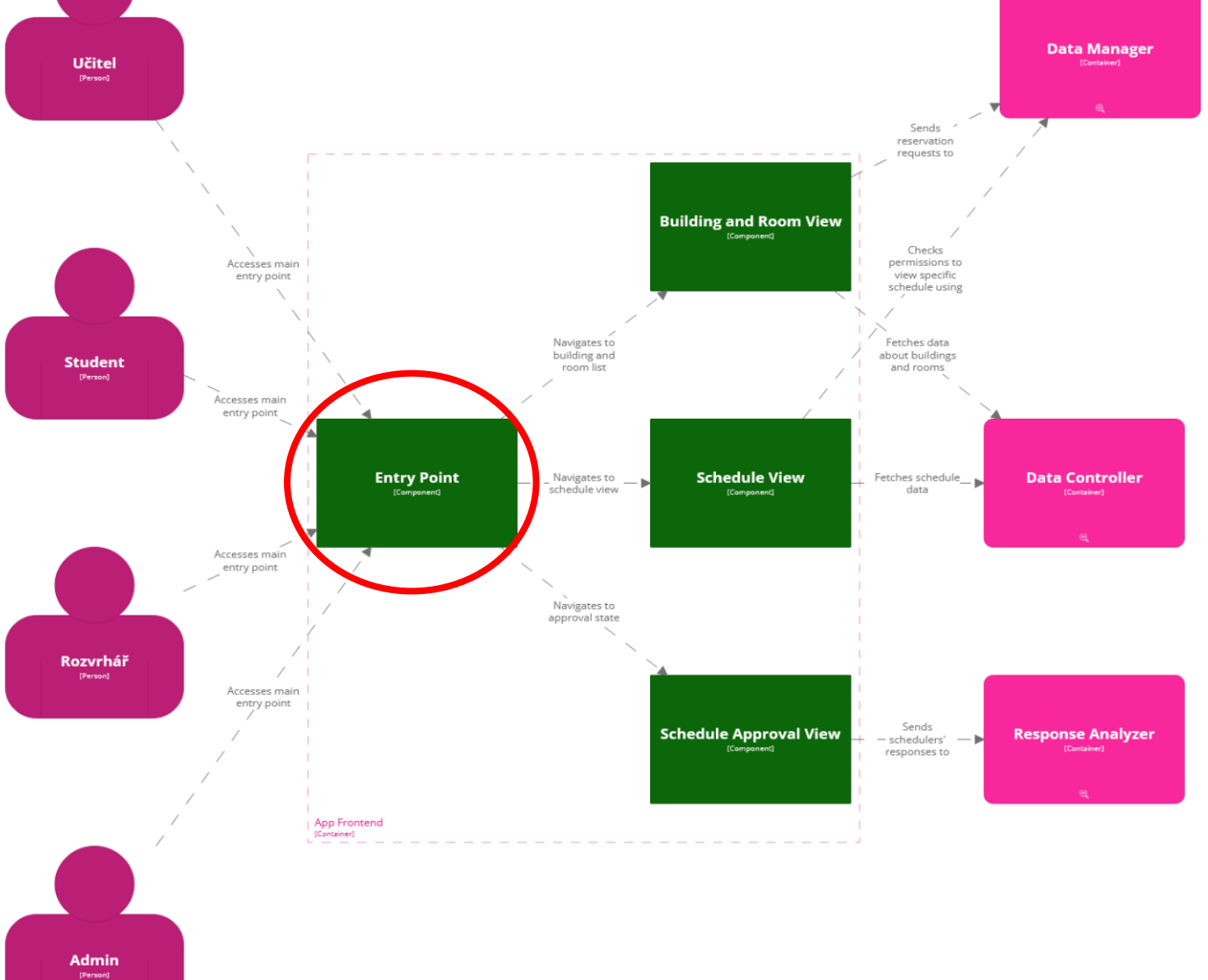
- *Source*: **Attacker**/Hacker
- *Stimulus*: wants **student** and **school data**
- *Artifact*: **AppFrontend** (to Data Controller)
- *Response*: Attacker's **script**/hack is **not accepted** and is **blocked** from the system
- *Measure*: The **data is untouched**

Architecture update

- Based on this scenario, the **AppFrontEnd container** should have an **extra "Entry point" component** that deals with script injections and unauthorised access.

  Also makes the diagram more readable.

Šimon



new

App_Frontend

Quality requirement scenario 2 – App Frontend

Design-time dimension - Modifiability

- *Source*: **App Frontend** - Building and Room View
- *Stimulus*: **New rules** for reservations are to be added
- *Artifact*: **Data Manager**
- *Response*: **New reservation rules** are **added** and working
- *Measure:* 5 man-day of testing

Architecture update

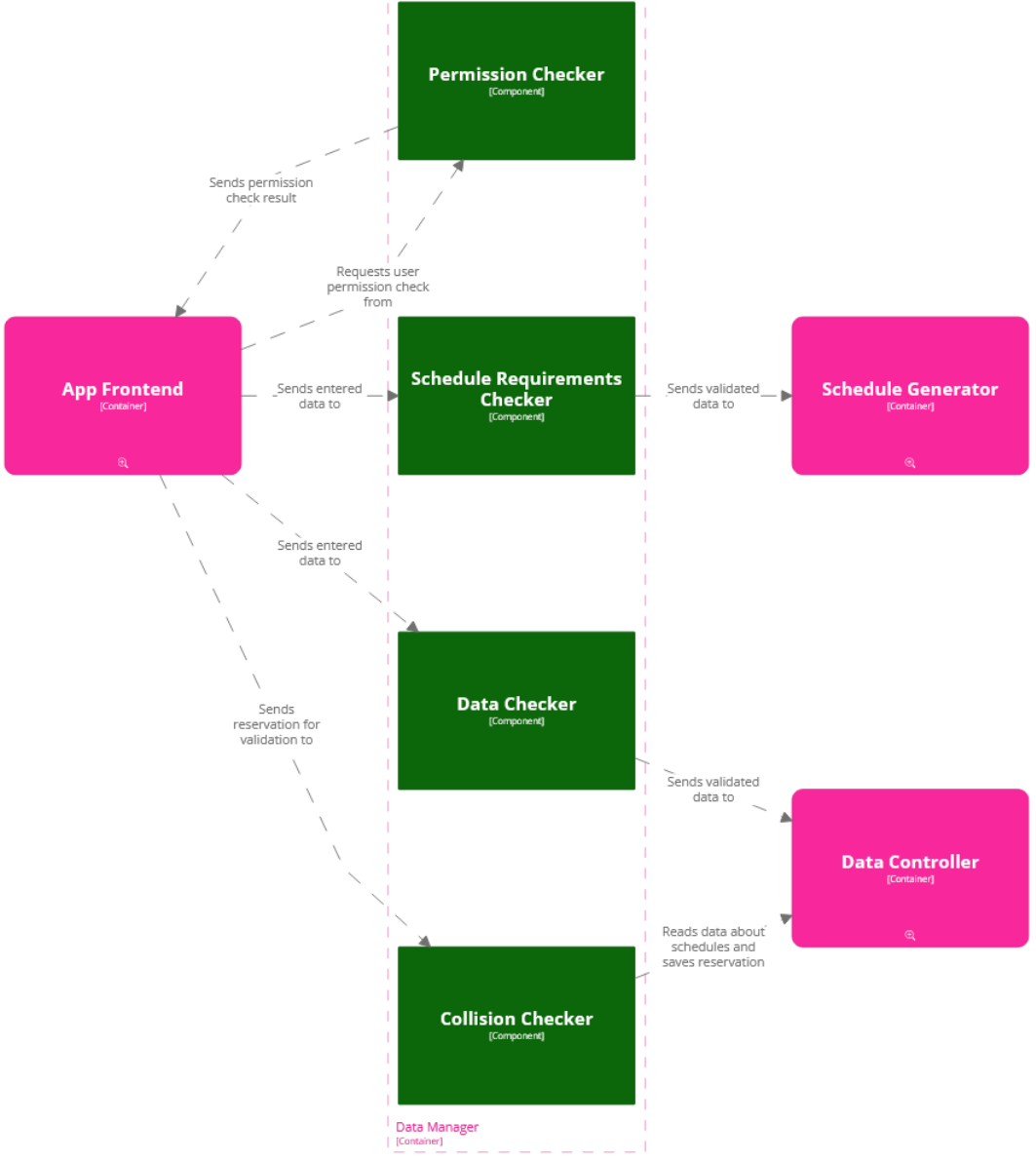
- The architecture **should be able to handle it.**

  Container - Data Manager contains neccessary "checker" components.

  A more localized approach may be beneficial to simplify this process.

Rastislav

Data Manager



Quality requirement scenario 3 – Data Manager

Run-time dimension - Availability

- *Source:* **App frontend**
- *Stimulus*: **sends wrong data** to validate
- *Artifact:* **Data manager** - schedule requirements/data checker
- *Response*: **informs the app frontend**
- *Measure:* 1 second

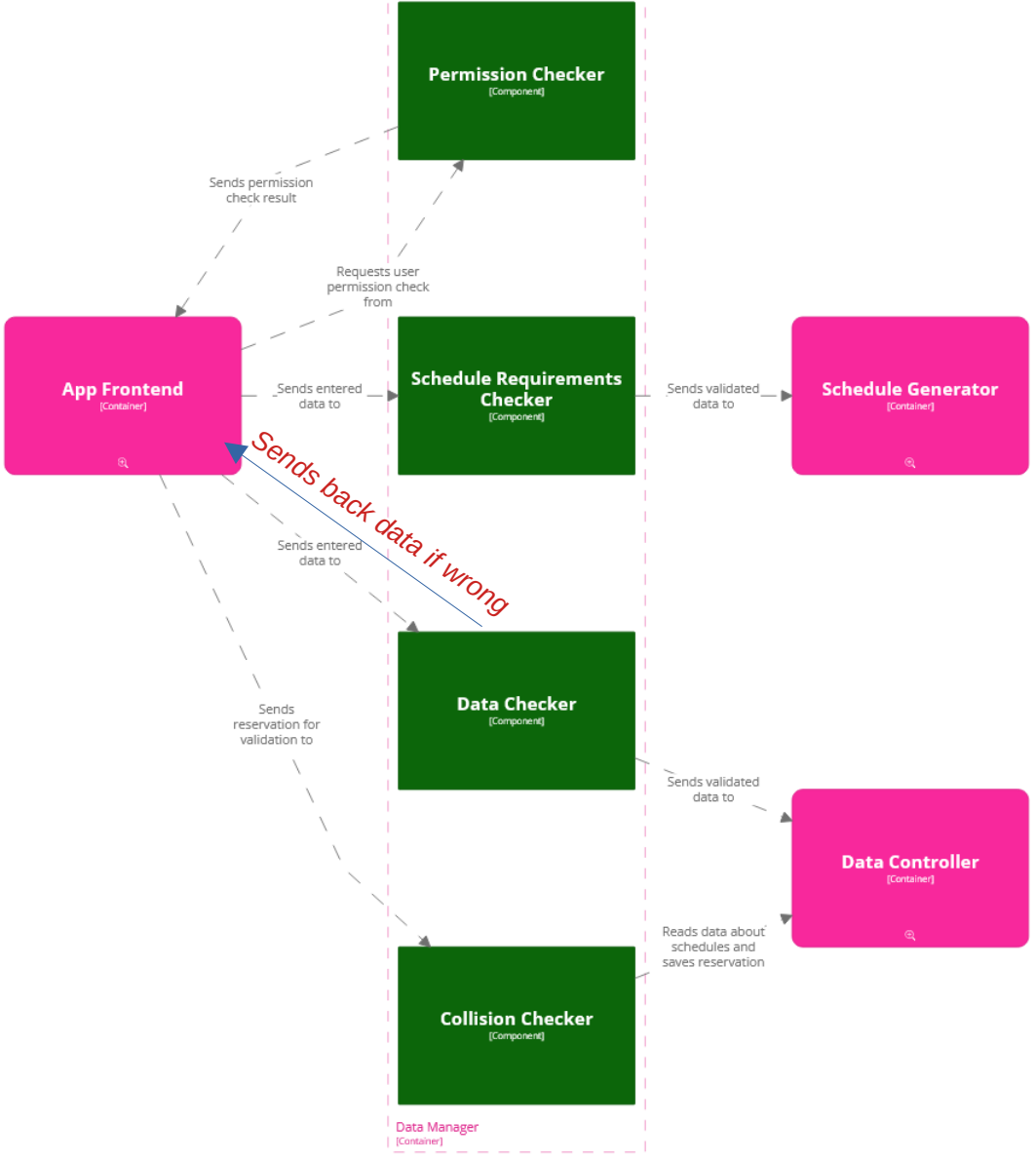
Architecture update

- The architecture **should inform app frontend** about what was **wrong with input data.**

  Create two-way communication between **App Frontend** and **Datachecker** component

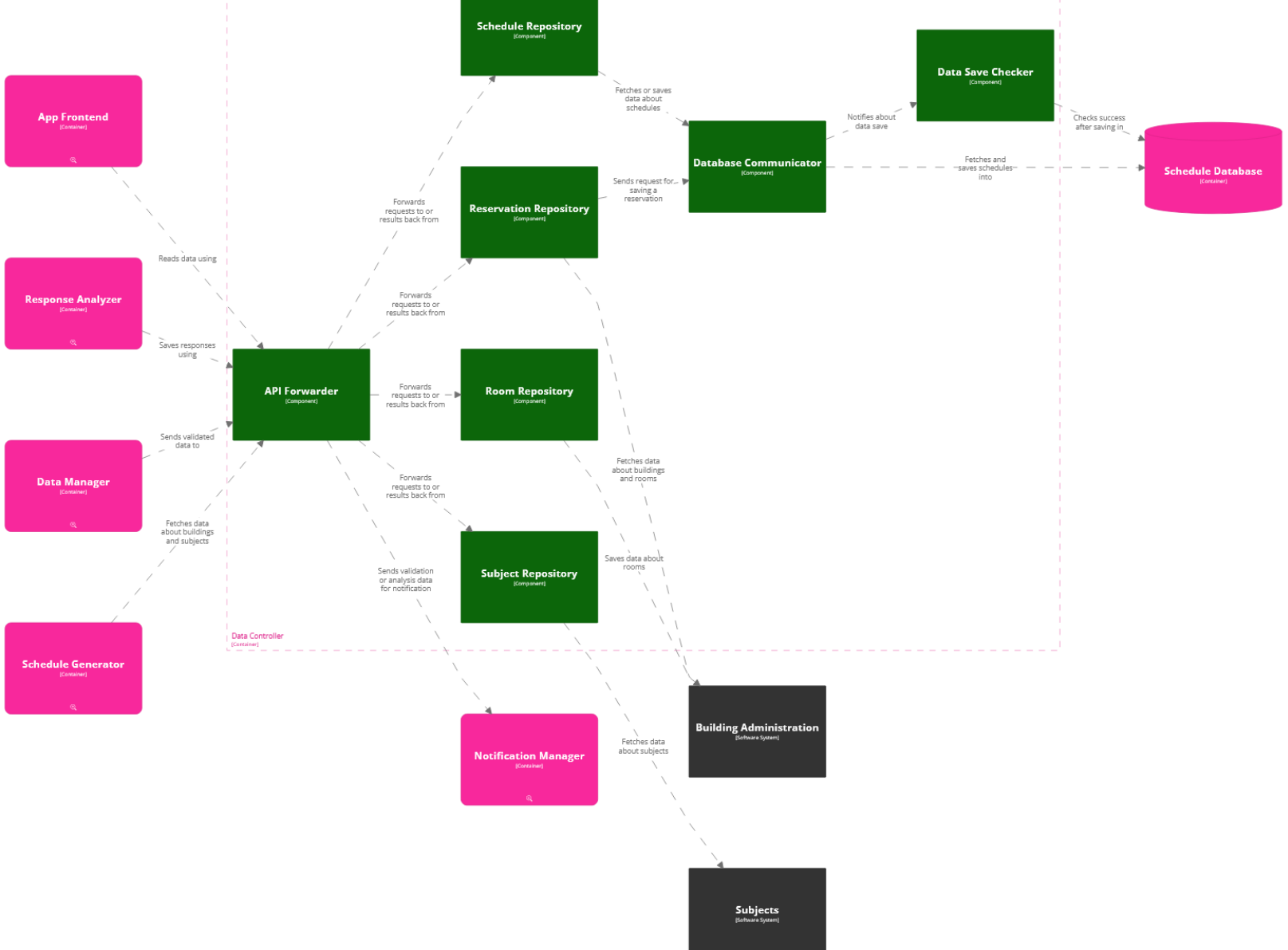
  or consider expanding architecture by component "info sender" that would better communicate with front end.

Matyáš



new

Data Manager



Data Controller

Quality requirement scenario 4 – Data Controller

Run-time dimension - Performance

- *Source:* **Student**
- *Stimulus*: **Displays information** about **buildings** and **schedules** (500 requests per minute)
- *Artifact:* **Data controller**
- *Response:* All **requests processed**
- *Measure*: With an average latency of 500 ms

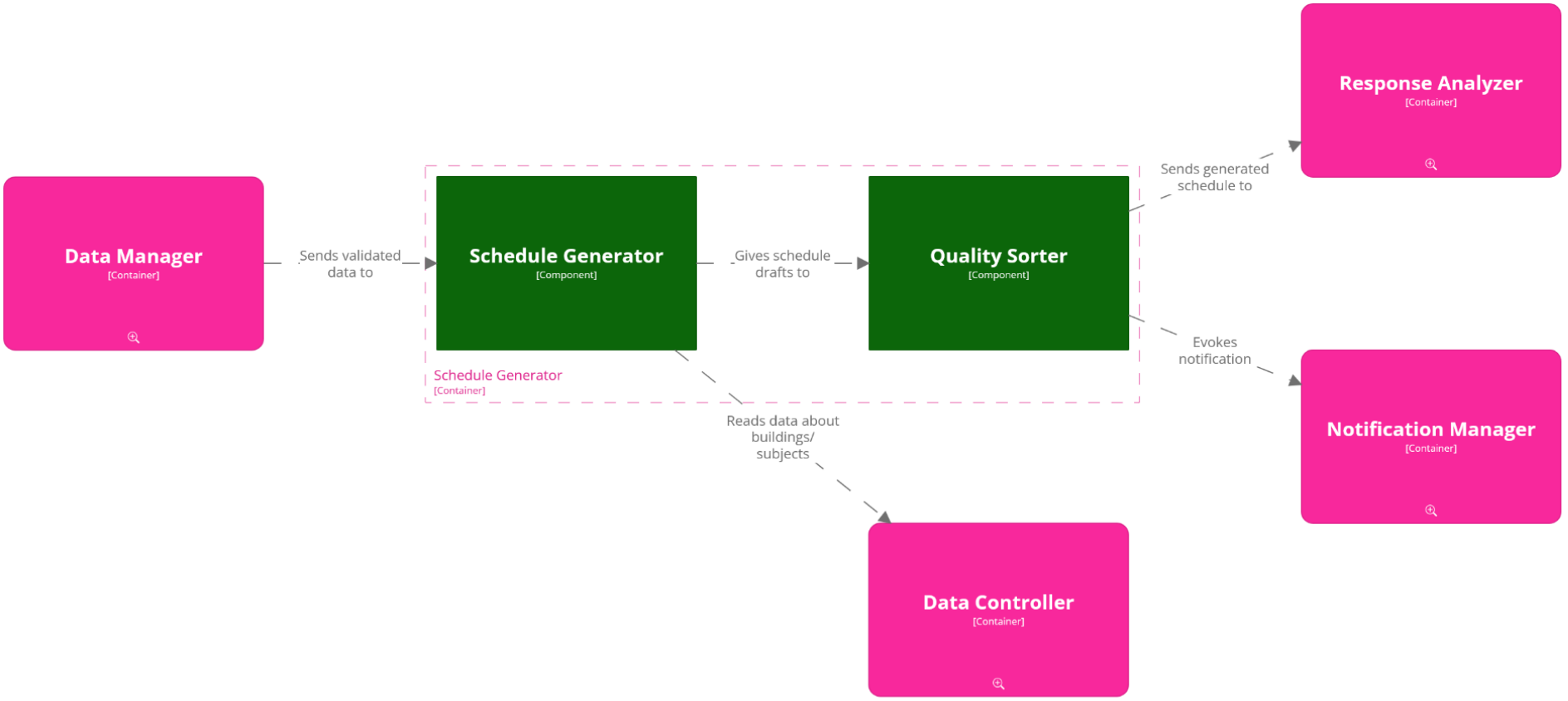
Architecture update

- The architecture **should be able to handle it.**

  All request are handled by data controller succesfully, so there is **no need for a change.**

Vojtěch

Schedule Generator



Quality requirement scenario 5 – Schedule Generator

Run-time dimension - Performance

- *Source:* **Schedule Generator -** Generator Component
- *Stimulus*: **Generated schedule** for **quality evaluation**
- *Artifact*: **Quality Sorter Component**
- *Response*: **Apply sorting rules** and **refine generated schedule**
- *Measure*: 5 seconds

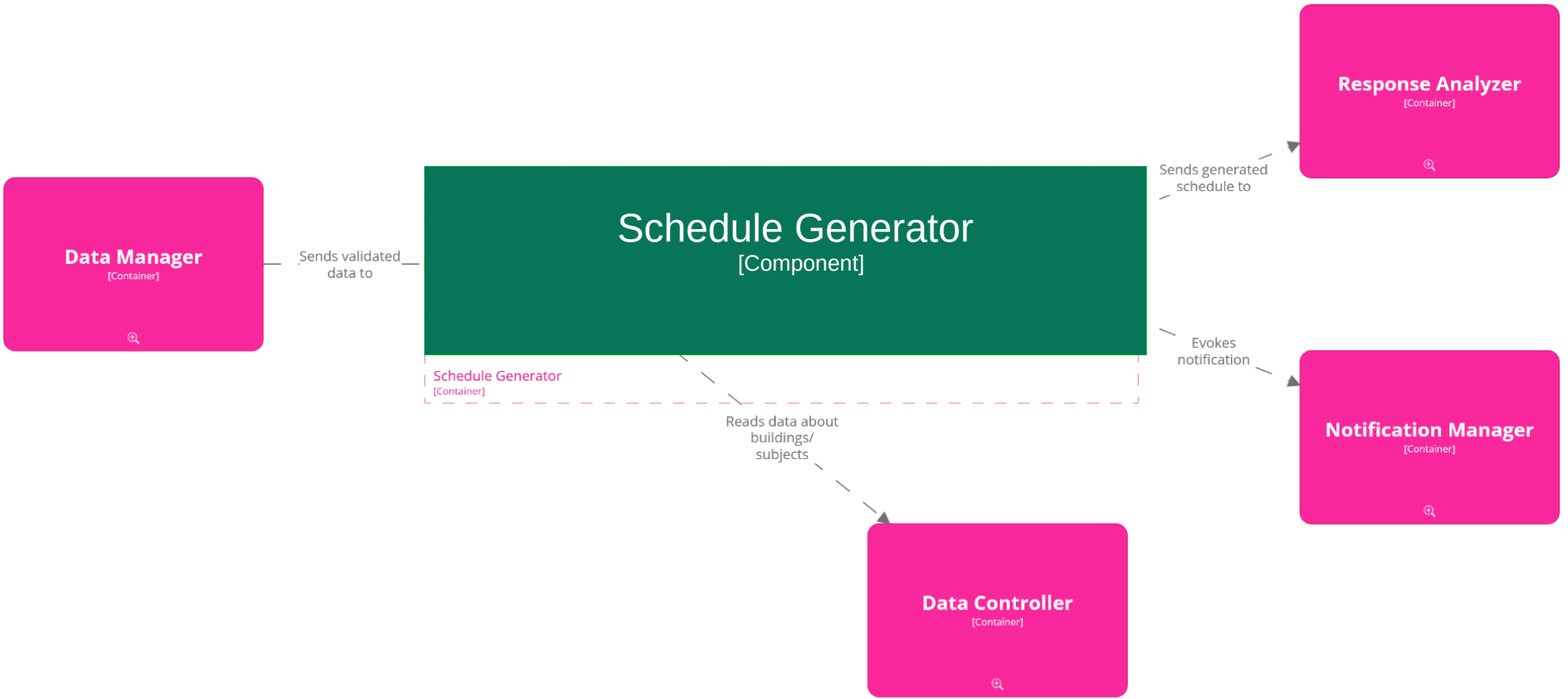
Architecture update

- The architecture should already apply sorting rules when creating the schedule, which can be resolved **by merging the two components.**

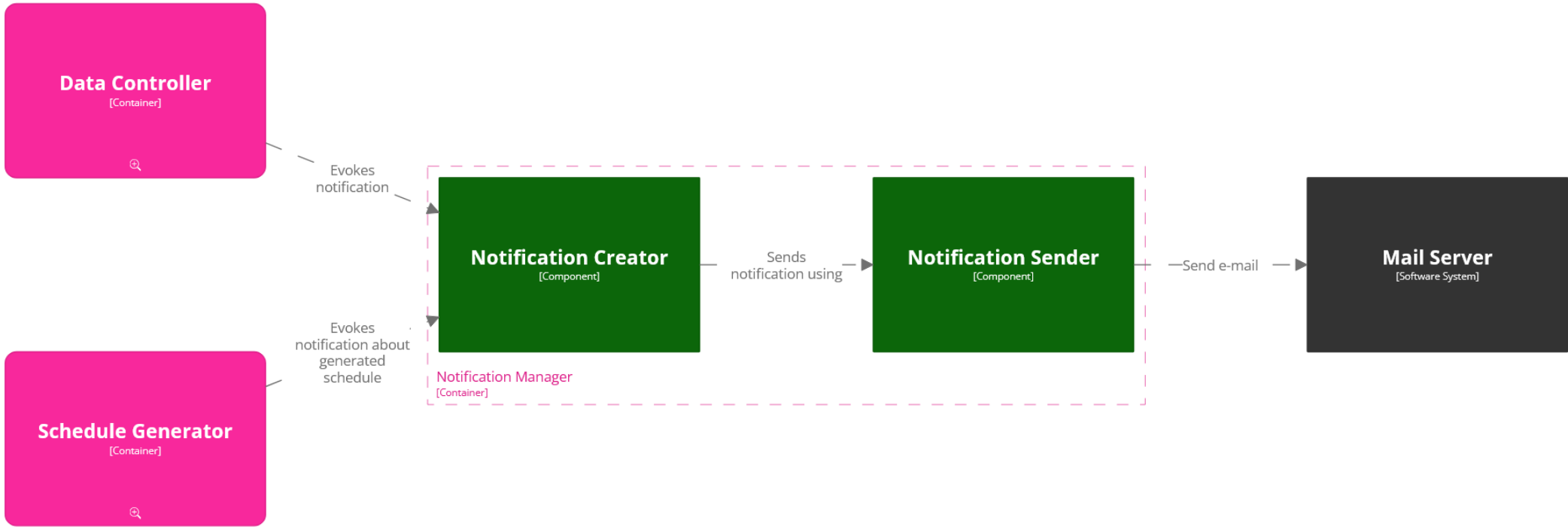
  This will lead to faster scheduling than doing 2 operations separately - generating and sorting.

Jan R.

NEW Schedule Generator



Notification Manager



Quality requirement scenario 6 – Notification Manager

Design-time dimension - Modifiability

- *Source:* **Notification Manager**
- *Stimulus*: **New mail server** to be used
- *Artifact:* Mail **server**
- *Response:* **New mail server is added**
- *Measure*: 1 man-day of testing

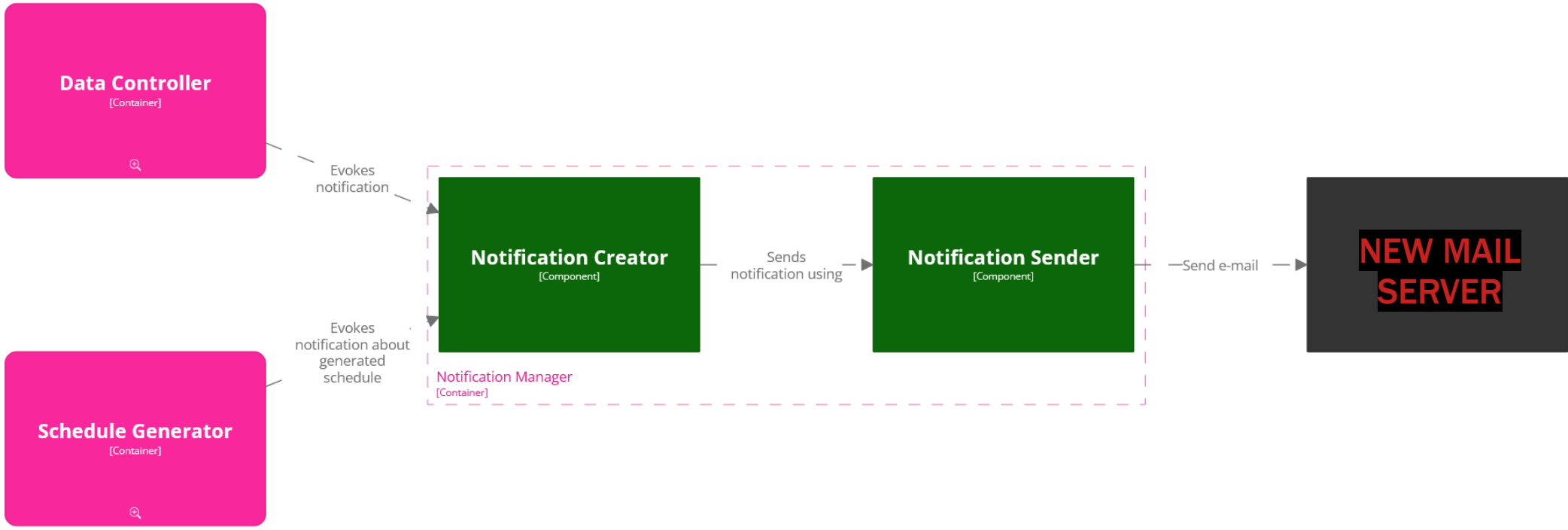
Architecture update

- The architecture **should be able to handle it.**

  Notifications are just sent to a different mail server.

Jan Č.

Notification Manager



Thank you for your attention!